NPS-MA-93-014

NAVAL POSTGRADUATE SCHOOL

Monterey, California

FINITE ELEMENT APPROXIMATION OF THE SHALLOW WATER EQUATIONS ON THE MASPAR

by

Beny Neta
Rex Thanakij

Technical Report For Period
November 1992 - March 1993

Prepared for: Naval Postgraduate School
Monterey, CA 93943

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

Rear Admiral T.A. Mercer
Superintendent

Harrison Shull
Provost

This report was prepared in conjunction with research conducted for the Naval Postgraduate School and funded by the Naval Postgraduate School.

This report was prepared by:

Beny Neta
Professor of Mathematics

Rex Thanakij

Reviewed by:

RICHARD FRANKE
Chairman

Released by:

PAUL J. MARTO
Dean of Research

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release Distribution unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS-MA-93-014	5 MONITORING ORGANIZATION REPORT NUMBER(S) NPS-MA-93-014
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable) MA	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943
8a NAME OF FUNDING/SPONSORING ORGANIZATION Naval Postgraduate School; 8b OFFICE SYMBOL (If applicable) MA	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER OM & N
8c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification) Finite Element Approximation of the Shallow Water Equation on the MASPAR

12 PERSONAL AUTHOR(S)
Beny Neta and Rex Thanakij

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical	FROM 11-92 TO 3-93	93-04-01	20

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	finite element approximation, shallow water equations

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Here we report on development of a high order finite element code for the solution of the shallow water equations on the massively parallel computer MP-1104. We have compared the parallel code to the one available on the Amdahl serial computer. It is suggested that one uses a low order finite element to reap the benefit of the massive number of processors available.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Beny Neta	22b TELEPHONE (Include Area Code) 408-656-2235; 22c OFFICE SYMBOL MA/Nd

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

FINITE ELEMENT APPROXIMATION OF THE SHALLOW WATER EQUATIONS ON THE MASPAR

Beny Neta
Naval Postgraduate School
Department of Mathematics
Code MA/Nd
Monterey, CA 93943
and
Rex Thanakij
MASPAR Computer Corporation
749 N. Mary Ave.
Sunnyvale, CA 94086

January 20, 1993

Abstract

Here we report on development of a high order finite element code for the solution of the shallow water equations on the massively parallel computer MP-1104. We have compared the parallel code to the one available on the Amdahl serial computer. It is suggested that one uses a low order finite element to reap the benefit of the massive number of processors available.

1 Introduction

The shallow water equations are first order nonlinear hyperbolic partial differential equations having many applications in Meteorology and oceanography. These equations can be used in studies of tides and surface water run-off. They may also be used to study large-scale waves in the atmosphere and ocean if terms representing the effects of the Earth's rotation are included. See review article by Neta (1992).

Indeed, it had become customary, in developing new numerical methods for weather prediction or oceanography, to study first the simpler nonlinear shallow water equations, which possess the same mixture of slow and fast waves as the more complex baroclinic three-dimensional primitive equations. One of the issues associated with the numerical solution of the shallow water equations is how to treat the nonlinear advective terms (Cullen and Morton, 1980, Navon, 1987). In this paper the two-stage Galerkin method combined with a high accuracy compact approximation to the first derivative is used. The method was developed by Navon (1987). See also Navon (1979_a, 1979_b, 1983). Our work here is to discuss porting issues of finite element onto a massively parallel machine. Section 2 discusses the algorithm, section 3 discusses the MasPar hardware and software. In section 4 we detail our numerical experiments and compare the results to the code running on the Amdahl serial computer.

2 Finite Element Solution

The barotropic nonlinear shallow-water equations on a limited-area domain of a rotating earth (using the β-plane assumption) have the following form:

$$\begin{aligned} u_t &+ uu_x + vu_y + \varphi_x - fv = 0 \\ v_t &+ uv_x + vv_y + \varphi_y + fu = 0 \qquad 0 \le x \le L,\ 0 \le y \le D,\ t > 0 \\ \varphi_t &+ (\varphi u)_x + (\varphi v)_y = 0\,. \end{aligned}$$

Here u and v are the velocity components in the x and y directions respectively, f is the Coriolis parameter approximated by the β plane as

$$f = f_0 + \beta(y - \frac{D}{2}),$$

A-1

where β, f_0, are constants and $\varphi = gh$ is the geopotential height. Periodic boundary conditions are assumed in the x direction and rigid boundary conditions ($v = 0$) are imposed in the y-direction. The domain is a cylindrical channel simulating a latitude belt around the earth (see e.g. Hinsman, 1975). The finite element approximation leads to systems of ODES which can be finite differenced in time (see e.g. Douglas and Dupont, 1970). In the two stage Galerkin (originally proposed by Cullen, 1974), we let any of the 4 derivatives in the nonlinear terms be approximated by the compact Numerov scheme, i.e. for

$$z_{xu} = \frac{\partial u}{\partial x}$$

we have

$$\frac{1}{70}[z_{i+2} + 16z_{i+1} + 36z_i + 16z_{i-1} + z_{i-2}] =$$
$$\frac{1}{84h}[-5u_{i-2} - 32u_{i-1} + 32u_{i+1} + 5u_{i+2}]$$

Similarly for z_{xv}, z_{yu} and z_{yv}.The approximation of $\frac{\partial v}{\partial x}$ requires an interpolation of the boundary values v_0, v_{N+1}

$$v_0 = 4v_1 - 6v_2 + 4v_3 - v_4$$
$$v_{N+1} = 4v_N - 6v_{N-1} + 4v_{N-2} - v_{N-3}$$
$$\left.\frac{\partial v}{\partial y}\right|_1 = \frac{-25v_1 + 48v_2 - 36v_3 + 16v_4 - 3v_5}{12h}$$
$$\left.\frac{\partial v}{\partial y}\right|_N = \frac{3v_{N-4} - 16v_{N-3} + 36v_{N-2} - 48v_{N-1} + 25v_N}{12h}$$

This stage will require a solution of a pentadiagonal system. For the second stage, we let w be any of the four nonlinear terms and we solve a tridiagonal system. For

$$w = vz$$

we have

$$\frac{1}{6}(w_{j-1} + 4w_j + w_{j+1}) = \tfrac{1}{12}(v_{j-1}z_{j-1} + v_j z_{j-1} + v_{j-1}z_j + v_{j+1}z_j + v_j z_{j+1} + v_{j+1}z_{j+1} + 6v_j z_j)$$

This two stage approximation yields $O(h^8)$ approximation to the derivatives u_x,u_y,v_x and v_y.

Now the approximation of the shallow water equations becomes

$$M(u_j^{n+1} - u_j^n) + \Delta t[(uz_{xu})_j^* + (vz_{yu})_j^* - f_j v_j^*] = \Delta t\overline{K}_{21}$$

$$M(v_j^{n+1} - v_j^n) + \Delta t[(vz_{yv})_j^* + u_j^{n+1}(z_{xv})_j + f_j u_j^{n+1}] = \Delta t\overline{K}_{31}$$

$$M(\varphi_j^{n+1} - \varphi_j^n) - \frac{1}{2}\Delta t K_1(\varphi_j^{n+1} + \varphi_j^n) = 0$$

where

$$\begin{aligned}
\overline{K}_{21} &= \frac{1}{2}(K_{21}^{n+1} + K_{21}^n) \\
\overline{K}_{31} &= \frac{1}{2}(K_{31}^{n+1} + K_{31}^n) \\
M_{ij} &= \iint_A V_j\, V_i\, dA \\
K_{1ij} &= \sum_k \iint_A V_i\, V_k u_k^* \frac{\partial V_j}{\partial x} dA + \sum_k \iint_A V_i\, V_k v_k^* \frac{\partial V_j}{\partial y} dA \\
K_{21}^{n+1} &= \sum_k \iint_A \varphi_k^{n+1} \frac{\partial V_k}{\partial x} V_i\, dA \\
K_{31}^{n+1} &= \sum_k \iint_A \varphi_k^{n+1} \frac{\partial V_k}{\partial y} V_j\, dA \\
K_{21}^n &= \sum_k \iint_A \varphi_k^n \frac{\partial V_k}{\partial x} V_i\, dA \\
K_{31}^n &= \sum_k \iint_A \varphi_k^n \frac{\partial V_k}{\partial y} V_j\, dA
\end{aligned}$$

and where V_i are the finite element shape functions.

$$u^* = u^{n+1/2} = \frac{3}{2}u^n - \frac{1}{2}u^{n-1} + O\left((\Delta t)^2\right)$$

and similarly for v^*.

Schuman (1957) filter was applied every 12 time steps to the v component of velocity in order to recover the higher accuracy of the method.

Since the two-stage Galerkin method does not conserve integral invariants (Cullen [1979]) we apply an aposteriori technique using an augmented Lagrangian nonlinearly constrained optimization approach for enforcing the conservation of integral invariants of the shallow water equations (see Navon and deVilliers (1983) and Navon (1983)).

3 System Overview

The MasPar family of massively parallel processing systems consists of arrays of 1K to 16K processing elements (PE), a scalar control unit (ACU) and a UNIX subsystem. Architecturally, each PE is a custom 64-bit RISC processor with 48 32-bit registers and 64 KB of data memory. All PEs execute instructions which are broadcast from the ACU on data stored in their local memory. Although there is only a single instruction stream, the processors have a number of autonomies, including the ability to generate independent addresses for indirect loads and stores to memory.

The PEs share data using two communication mechanisms: the xnet and the router. The xnet is an eight-way nearest neighbor mesh that is used for structured communications such as stencil operations in finite difference codes. The router is a multi-stage circuit-switched network for global or random communication patterns. I/O to and from the PEs is transferred via the router to an external memory buffer called I/O RAM. From I/O RAM, data can asynchronously be transferred to a wide variety of devices such as disk arrays, frame buffers, or other machines. The MasPar Disk Array (MPDA) provides up to 22 GB of formatted capacity as a true UNIX file system. The UNIX subsystem provides the programming and run-time environment to users.

3.1 MasPar Software

The MasPar system is programmed in either MPL, a parallel extension to ANSI C, or MasPar Fortran, an implementation of Fortran 90. In MasPar Fortran (MPF) parallel operations are expressed with the Fortran 90 (F90) array extensions which treat entire arrays as manipulatable objects, rather than requiring them to be iterated through one element at a time. F90 has also added a significant number of intrinsic libraries; operations such as matrix multiplication and dot product are part of the language. Since Fortran 90 is a standard defined by the ANSI/ISO committees, programs are architecture independent and can be transparently moved to other platforms.

Fortran 77	*Fortran 90*
$do\, i = 1, 256$ $\quad do\, j = 1, 256$ $\quad\quad a(i,j) = b(i,j) + c(i,j)$ $\quad enddo$ $enddo$	$a = b + c$

The Fortran 90 code can be run on any computer with a F90 compiler. On a scalar machine such as a workstation, the arrays will be added one element at a time; just as if it had been written in Fortran 77. On a vector machine, the number of elements added at a time is based on the vector length; a machine with a vector length of 64 will add 64 array elements at once. The MasPar machine acts like a vector machine with a very long vector. On a 16K MasPar machine, 16384 arrays elements are added simultaneously.

MasPar provides key routines in math, signal, image, and data display libraries. The Math Library (MPML) contains a number of high-level linear algebra solvers, including a general dense solver with partial pivoting, a Cholesky solver, a conjugate solver with preconditioning, and an out-of-core solver. MPML also includes a set of highly-tuned linear algebra building blocks, analogous to BLAS on vector machines, from which the user can develop additional solvers. The Data Display Library provides a convenient interface to graphically display data from within a program as it is executing.

The MasPar Programming Environment (MPPE) is an integrated, graphical environment for developing, debugging, and tuning applications. MPPE provides a rich set of graphical tools that allow the user to interactively control and visualize a program's behavior. The

statement level profiler allows the user to quickly identify the compute-intensive sections of the program while the machine visualizer details the use of hardware resources. Each of these tools are continuously available without having to recompile, even if a program has been compiled with optimizations.

4 Program

The program is modular and is complemented with easily reachable switches controlling print and plot options. The Input to the program consists of a single line containing the following six parameters:

DT - the time step in seconds (F5.2)

NLIMIT - total number of time steps (I5)

MF - number of time steps between printing solution (I5)

NOUTU - to print (1) or not to print (0) the u-component

NOUTV - to print (1) or not to print (0) the v-component

NPRINT - to print (1) or not to print (0) the global nodal numbers of each triangular elements and the indices and node coordinates of the nonzero entries of the global matrix.

The main program initializes all variables and then reads the only data card of the program. It then proceeds to index and label the nodes and the elements, thus setting up the integration domain. This is done by subroutine NUMBER.

Subroutine CORRES determine the nonzero locations in the global matrix and stores them in array LOCAT. The initial fields of height and velocity are set up by subroutine INCOND. The derivatives of the shape functions (V_j) are calculated in AREAA. A compact storage scheme for the banded and sparse global matrices is implemented in subroutine ASSEM. The method is based on the fact that the maximum number of triangles supporting any node is six. Three different types of element matrices (3 x 3) will be required for assembly in the global matrices.

A switch, denoted NSWITCH is set for selecting between the different types of element matrices. After setting up the time independent global matrices the program proceeds to the main do-loop which performs the time-integration and which is executed once for every new time-step.

As the solution of the nonlinear constrained optimization problem of enforcing conservation of the nonlinear integral invariants requires scaling of the variables, the scaling is performed in the main program as well as in subroutine INCOND.

In the main integration loop the simulation time is set up and adjusted and then the subroutines ASSEM and MAMULT set up and assemble the global matrices which then are added up in a matrix equation, first for the continuity equation and in a similar manner for the u and v-momentum equations.

Subroutine SOLVER then is called to solve the resulting system of linear equations (of block tridiagonal form) by the conjugate gradient square.

The new field values for the geopotential and velocities, $\phi_{ij}^{n+1}, u_{ij}^{n+1}, v_{ij}^{n+1}$ respectively, are used immediately as obtained in solving the coupled shallow-water equations system. For the u and v-momentum equations, the new two-stage Numerov-Galerkin scheme is imple-

mented. Separate routines are set up for the x and y-derivatives advection terms, DX and DY respectively. Subroutine DX implements the two-stage Numerov-Galerkin algorithm described previously for the advective terms in the u and v-momentum equations involving the x-derivative.

In the first stage it calculates the $O(h^8)$ accurate generalized-spline approximation to the $(\partial u/\partial x)$ first derivative by calling upon subroutine CYCPNT which solves a periodic pentadiagonal system of linear equations generated by the spline approximation.

In the second stage it implements the second part of the Numerov-Galerkin algorithm for the nonlinear advective term $u(\partial u/\partial x)$ and solves a cyclic tridiagonal system by calling upon subroutine CYCTRD. Subroutine DY implements the two-stage Numerov-Galerkin algorithm described previously for the advective terms in the u and v-momentum equations involving the $y-$ derivative. In its first stage it calculates the $O(h^8)$ accurate generalized-spline approximation to the $(\partial u/\partial y)$ first derivative by calling upon subroutine PENTDG which solves the usual pentadiagonal system of linear equations generated by the generalized-spline approximation.

In the second stage subroutine DY implements the second part of the Numerov-Galerkin algorithm for the nonlinear advective term $u(\partial u/\partial y)$ and solves the Galerkin product by calling upon subroutine NCTRD to solve a special tridiagonal system.

The boundary conditions are implemented by subroutine BOUND. Periodically, a Schuman filtering procedure is implemented for the v-component of velocity only, by calling subroutine SMOOTH. The integral invariants are calculated at each time-step by calling subroutine LOOK. If the variations in the integral invariants exceed the allowable limits δ_E, δ_H, or δ_Z, the Augmented-Lagrangian nonlinear constrained optimization procedure is activated. The unconstrained optimization uses the conjugate-gradient subroutine E14DBF of the NAG(1982) scientific library. Subroutine E14DBF calls a user-supplied subroutine FUNCT which evaluates the function value and its gradient vector as well as subroutine MONIT whose purpose is merely to print out different minimization parameters.

After a predetermined number of steps, subroutine OUT is called, which in turn calls upon the subroutines LOOK to calculate the integral invariants. Practically 4-5 augmented-Lagrangian minimization cycles were determined to be sufficient.

We ran the program under MPPE and the following table shows the CPU time used by some of the routines. All others require less than 5% each. Therefore we have decided to parallelize ASSEM, MAMULT, SOLVER (switching from Gauss Seidel to Conjugate Gradient Square). Other subroutines we parallelized are:

CORRES, INCOND, LOOK, MONIT, NUMBER and AREAA.

After this, the most time consuming routines become E14DBF and FUNCT. These are required only if the integral constraints are not conserved. Therefore if the mesh is fine, these routines will not be called. Our numerical experiments confirmed that these two routines were called only in the coarsest grid case.

The next set include: DX, DY, CYCTRD, CYCPNT, NCTRD, PENTDG, TRIDG, and SMOOTH. We have decided not to try at this point to parallelize these or BOUND. We have ran this program on the MP-1104 (4096 processors) on a variety of grid sizes. The

Routines	CPU
SOLVER	32%
ASSEM	25%
MAMULT	14%
CORRES	5%
BOUND	5%

Table 1: CPU time used by some routines

original program was also ran on the Amdahl 5990/500 serial computer. All computations were performed in double precision. The domain is a rectangle 6000 km by 4400 km. The coarsest mesh, $\Delta x = \Delta y = 400km$. This means that the number of grid points in the x-direction, NC, is 15 and the number of grid points in the y-direction, NROW is 11. (Δt will be adjusted for stability.) The number of time steps, NLIMIT, is 30.

NC	NROW	$\Delta x(km)$	$\Delta y(km)$	Δt(sec)	Amdahl (sec)	MP-1104(sec)
15	11	400	400	18.	1.14	14
48	45	$133\frac{1}{3}$	$133\frac{1}{3}$	5.51	13.52	31.3
63	62	93.75	70.97	4.22	24.8	44.3
88	85	51.76	51.76	3.03	48.32	80
128	125	46.87	46.87	2.10	-	164

Table 2: Total CPU time for several grids

The initial condition for the height field is given by

$$h(x,y) = H_0 + H_1 \tanh \frac{9(D/2-y)}{2D} + \frac{H_2}{\cosh^2 \frac{9(D/2-y)}{2D}} \sin \frac{2\pi x}{L}$$

where

$$H_0 = 2000m, \qquad H_1 = -220m, \qquad H_2 = 133m,$$

and

$$f_0 = 10^{-4} sec^{-1}, \qquad \beta = 1.5 \times 10^{-11} sec^{-1} m^{-1}.$$

This initial condition is given in Grammeltveldt (1969) and tested by several researchers (Cullen and Morton (1980), Gustafsson (1971), Navon (1987) etc.) The initial velocity fields were derived from the initial height field via the geostrophic relationships

$$u = -\frac{g}{f}\frac{\partial h}{\partial y}$$

$$v = \frac{g}{f}\frac{\partial h}{\partial x}.$$

Table 2 gives the CPU time for each grid.

If we compare the CPU time for three of the subroutines we parallelized (to avoid the difficulty that some parts are still running on the front end) we find that in MAMULT and SOLVER we were able to cut the CPU time. The results are summarized in Table 3.

Subroutine	Problem size	Amdahl (sec)	MP-1104 (sec)
ASSEM	48 by 45	3.02	5.77
	63 by 62	5.47	8.56
	88 by 85	10.49	15.2
	128 by 125	–	34.4
MAMULT	48 by 45	.42	.44
	63 by 62	.74	.37
	88 by 85	1.44	.88
	128 by 125	–	1.53
SOLVER	48 by 45	7.21	5.97
	63 by 62	13.14	4.87
	88 by 85	25.38	10.6
	128 by 125	–	17.9

Table 3: CPU time before and after parallelization

The code was ran under profiler and we found that now the CPU usage (in percent of total CPU) is as given in table 4.

It is clear that one should parallelize DX,DY,PENTDG,TRIDG and LOOK. The first four require that one parallelizes the subroutines NCTRD,CYCTRD and CYCPNT. This is not done since the tridiagonal and pentadiagonal systems to be solved are of order NC. We feel that one should approach this problem slightly differently. Instead of trying to parallelize this code which is of high order, we should parallelize a low order finite element code for the shallow water equations. The accuracy of the solution will be obtained by using an even finer mesh than 46 km (NC=128) we used above. It will be interesting to compare the accuracy and efficiency of the two codes on MP-1104 machine.

Subroutine	15 by 11	44 by 45	88 by 85	128 by 125
FUNCT	36.8	-	-	-
DX	3.2	12.3	17.0	18.6
DY	3.2	12.8	16.6	20.0
ASSEM	10.2	17.9	16.0	14.3
PENTDG	2.5	12.0	13.7	11.4
MAMULT	16.2	13.7	9.8	6.9
TRIDG	1.2	6.5	6.9	5.2
LOOK	9.1	4.1	4.4	8.4
NCTRD	.7	3.2	3.3	2.5
CYCPNT	.7	3.9	3.2	2.4
CYCTRD	.8	2.6	2.1	1.5
SOLVER	8.0	4.0	1.9	1.1
SET STI	1.0	1.7	1.4	1.2
BOUND	1.8	1.7	1.0	3.9
VFEUDX	1.8	1.3	.6	.5
rest	2.8	2.1	2.1	2 1

Table 4: CPU time by subroutine after parallelization

Conclusion

We have developed a high order finite element code to solve the shallow water equations on the MasPar massively parallel computer MP-1104. It is believed that a low order finite element code will be more efficient on the MP-1104 computer.

Acknowledgement

The first author would like to thank MasPar Computer Corporation for the computer time used to develop the code. This research was conducted for the Office of Naval Research and funded by the Naval Postgraduate School.

References

M.J.P. Cullen, A finite-element method for a nonlinear initial value problem, J. Institute of Mathematics and its Applications, **13** (1974), 233-247.

M.J.P. Cullen, The finite element method in "Numerical Methods Used in Atmosphere Models," Vol. 2, ICSU/WMO GARP Pub. Ser. No. 17, World Met. Org., Geneva, Switzerland, 1979.

M.J.P. Cullen and K.W. Morton, Analysis of Evolutionary error in finite-element and other methods, J. Computational Physics, **34** (1980), 245-267.

J. Douglas and T. Dupont, Galerkin methods for parabolic problems, SIAM J. Numerical Analysis, **7** (1970), 575-626.

D.E. Hinsman, Application of a finite-element method to the barotropic primitive equations, M. Sc. Thesis, Naval Postgraduate School, Department of Meteorology, Monterey, CA, 1975.

NAG, Numerical Algorithms Group Fortran Library Manuals Volumes 1-6 (1982) NAG, Banbury Road, Oxford, OX2-6HN, England or NAG - Inc. 1131 Warren Ave. Downers Grove IL 70515

I.M. Navon, *Finite-element simulation of the shallow-water equations model on a limited* area domain, Applied Mathematics and Modeling, **3** (1979), 337-348.

I.M. Navon, Finite-element solution of the shallow-water equations on a limited area domain with three different mass matrix formulations, Proceeding of the 4^{th} Conference on Numerical Weather Prediction, Silver Springs, MD, 1979, 223-227.

I.M. Navon, A Numerov-Galerkin technique applied to a finite-element shallow-water equations model with enforced conservation of integral invariants and selective lumping, J. Computational Physics, **52** (1983), 313-339.

I.M. Navon and R. de Villiers, Combined penalty-multiplier optimization methods to enforce integral invariants conservation, Monthly Weather Review, **111** (1983), 1228-1243.

I.M. Navon, FEUDX: A two-stage, high-accuracy, finite-element Fortran program for solving shallow-water equations, Computers and Geosciences, **13** (1987), 255-285.

B. Neta,, Analysis of Finite Elements and Finite Differences for Shallow Water Equations: A Review, Mathematics and Computers in Simulation, **34** (1992), 141-162.

F.G. Schuman, Numerical methods in weather prediction II, smoothing and filtering, Monthly Weather Review, **85** (1957), 357-361.

```
        SUBROUTINE ASSEM (COMA,STI,NSWTCH,CODI,AREA,tnod,tlocat)
        INCLUDE     'PAR'
        include     'inter_info'
        real        COMA(7,NODEB),STI(NNOD,NNOD,NELE)
        real        CODI(NNOD,NELE)
        integer     itemp, t1, t2, t3, t4, t0, time1, time2
        real        tgmat(7,NELE)
        integer     tlocat(6,NODE), tnod(NNOD,NELE), nswtch

CMPF  ONDPU STI,CODI,COMA
cmpf  map coma(memory,allbits)
cmpf  map codi(memory,allbits)
cmpf  map tgmat(memory,allbits)
cmpf  map tlocat(memory,allbits)
cmpf  map tnod(memory,allbits)
cmpf  map sti(memory,memory,allbits)

        COMA = 0.0
C
C    DECIDE WHICH ELEMNT MATRIX MUST BE CALCULATED

        GOTO (100,200,500,600), NSWTCH
C
100   continue
        call assem_s1(codi, tgmat, tlocat, tnod)
        coma(:,:) = tgmat(:,:NODEB)
        RETURN
C
200     continue

        write(6,*) ' error for nswitch = 2'
       RETURN
C
500   continue
        call assem_s2(area, tgmat, tlocat, tnod)
        coma(:,:) = tgmat(:,:NODEB)
       RETURN
C
600    continue
        call assem_s3(sti, tgmat, tlocat, tnod)
        coma(:,:) = tgmat(:,:NODEB)
return
       END
```

```
        subroutine assem_s1(codi, gmat, locat, nod)
        include 'PAR'
        real, intent(in)  :: codi(NNOD, NELE)
        real, intent(out) :: gmat(7,NELE)
        integer, intent(in) :: locat(6,NODE), nod(NNOD,NELE)
        real  tcodi(NNOD,NELE)
cmpf map codi(memory,allbits)
cmpf map gmat(memory,allbits)
cmpf map locat(memory,allbits)
cmpf map nod(memory,allbits)
cmpf map tcodi(memory,allbits)
        integer irow(NELE), icol(NELE), i, k, j, l

        gmat = 0.
        tcodi = codi/6.

        do 100 k = 1, NNOD
           irow = nod(k,:)
                do 150 j = 1, NNOD
                    icol = nod(j,:)
                    if( k .eq. j) then
cmpf collisions
                           gmat(7,irow)=gmat(7,irow)+tcodi(j,:)
                           goto 150
                    endif
                   do 200 l = 1, 6
                         where(locat(l,irow) .eq. icol)
cmpf collisions
                            gmat(l,irow) = gmat(l,irow) + tcodi(j,:)
                         end where
200                continue
150             continue
100       continue

        return
        end
```

```
        subroutine assem_s2(area, gmat, locat, nod)
        include 'PAR'

        real            area, tarea12, tarea6
        integer         locat(6,NODE), nod(NNOD,NELE)
        real          gmat(7,NELE)
cmpf map gmat(memory,allbits)
cmpf map locat(memory,allbits)
cmpf map nod(memory,allbits)
        integer irow(NELE), icol(NELE), i, k, j

        gmat = 0.
        tarea12 = area/12.
        tarea6 = tarea12 * 2.


        do 100 k = 1, NNOD
           irow = nod(k,:)
                  do 150 j = 1, NNOD
                     icol = nod(j,:)
                        if( k .eq. j) then
cmpf collisions
                          gmat(7,irow)=gmat(7,irow)+tarea6
                          goto 150
                        endif
                    do 200 l = 1, 6
                        where(locat(l,irow) .eq. icol)
cmpf collisions
                           gmat(l,irow) = gmat(l,irow) + tarea12
                        end where
200                 continue
150               continue
100      continue

         return
         end
```

```
      subroutine assem_s3(sti, gmat, locat, nod)
      include 'PAR'
      real, intent(out) ::  gmat(7,NELE)
      real, intent(in) :: sti(NNOD,NNOD,NELE)
      integer, intent(in) :: locat(6,NODE), nod(NNOD,NELE)
cmpf map gmat(memory,allbits)
cmpf map locat(memory,allbits)
cmpf map nod(memory,allbits)
cmpf map sti(memory,memory,allbits)
      integer irow(NELE), icol(NELE), i, k, j, l

      gmat = 0.
      do 100 k = 1, NNOD
         irow = nod(k,:)
             do 150 j = 1, NNOD
                icol = nod(j,:)
                 if( k .eq. j) then
cmpf collisions
                     gmat(7,irow)=gmat(7,irow)+sti(k,j,:)
                     goto 150
                 endif
               do 200 l = 1, 6
                   where(locat(l,irow) .eq. icol)
cmpf collisions
                       gmat(l,irow) = gmat(l,irow) + sti(k,j,:)
                   end where
200            continue
150          continue
100    continue

      return
      end
```

```
      SUBROUTINE MAMULT (COMA,VECTOR,RIGHT,locat)
      INCLUDE 'PAR'

real, intent(in) ::  COMA(7,NODEB), vector(:)
      real, intent(out) :: RIGHT(:)
       integer, intent(in) :: locat(6,NODE)
       integer  nloc(NODE)
       integer kr

cmpf map coma(memory,allbits)
cmpf map locat(memory,allbits)

       right = 0.
       RIGHT(:NODE) = coma(7,:node)*vector(:node)
       DO 80 KR=1,6
              nloc(:)=locat(kr,:)
              where(nloc(:).ne.0)
     &         RIGHT(:node) = RIGHT(:node) + COMA(KR,:node)*VECTOR(nloc(:))
80    CONTINUE
      RETURN
end

! Conjugate Gradient Square (CGS) method to solve non-symmetric
! positive definite metrix.   Ax = b
!
! coma = input matrix A
! right = b
! xsolv = x

       subroutine my_solver(coma,right,xsolv,eps,itermx,locat)
       include 'PAR'
       include 'mamult_if'

       real coma(7,NODEB), eps
       real right(NODEB), xsolv(NODEB)
       real,  dimension(NODE) :: r, rbar, p, p1, q, u, mv
       real*8,  dimension(NODE) :: brbar, bp1
       real beta, conv1, conv2, rest1, rest2
       real del0, del1, alpha, residual(100)
       real*8 bdel0, bdel1
       integer locat(6,NODE)
       integer itermx, i, j
       common/debug/ntime
```

```
cmpf map coma(memory, allbits)
cmpf map locat(memory,allbits)
        r = right(:node)
        conv1 = dotproduct(r,r)
        call mamult(coma, xsolv, mv, locat)
        r = r - mv
        rbar = r
        p = r
        u = r
        itermx = 100
        p1 = 0.
        eps = 1.5e-6
        do 10 i = 1, itermx
           call mamult(coma, p, p1, locat)
           del0 = dotproduct(rbar,p1)
           del1 = dotproduct(rbar,r)
           alpha = del1/del0
           q = u - alpha*p1
           u = u + q
           xsolv(:node) = xsolv(:node) + alpha*u
           call mamult(coma, u, p1, locat)
           del0 = del1
           r = r - alpha*p1
           conv2 = dotproduct(r,r)
           residual(i) = sqrt(conv2/conv1)
           if( residual(i) .lt. eps) then
                return
           endif
           del1 = dotproduct(rbar,r)
           beta = del1/del0
           u = r + beta*q
           p = u + beta * (q + beta*p)
10      continue

        PRINT 2001
2001    FORMAT (1X,'NO CONVERGENCE')
        do 20 i = 1, itermx
                 write(6, *) i, ' residual is ', residual(i)
20      continue
       stop
       END
```

DISTRIBUTION LIST

Director Defense Technology Information Center Cameron Station Alexandria, VA 22314	2
Director of Research Administration Code 012 Naval Postgraduate School Monterey, CA 93943	1
Library Code 0142 Naval Postgraduate School Monterey, CA 93943	2
Department of Mathematics Code MA Naval Postgraduate School Monterey, CA 93943	1
Center for Naval Analysis 4401 Ford Avenue Alexandria, VA 22302-0268	1
Professor Beny Neta Code MA/Nd Department of Mathematics Naval Postgraduate School Monterey, CA 93943	15
Professor Naotaka Okamoto Okayama University of Science Department of Applied Science Ridai-cho 1-1, Okayama 700 Japan	1

Professor William Gragg Code MA/Gr Department of Mathematics Naval Postgraduate School Monterey, CA 93943	1
Professor Levi Lustman NOARL Monterey, CA 93943	1
Dr. C.P. Katti J. Nehru University School of Computer and Systems Sciences New Delhi 110067 India	1
Professor Paul Nelson Texas A&M University Department of Nuclear Engineering and Mathematics College Station, TX 77843-3133	1
Professor I. Michael Navon Florida State University Supercomputer Computations Research Institute Tallahassee, FL 32306	1
Professor M.M. Chawla, Head Department of Mathematics III/III/B-1, IIT Campus Hauz Khas, New Delhi 110016 India	1
Professor M. Kawahara Department of Civil Engineering Faculty of Science and Engineering Chuo University Kasuga 1-chome 13 Bunkyo-ku, Tokyo Japan	1
Professor H. Dean Victory Jr. Texas Tech University Department of Mathematics Lubbock, TX 79409	1

Professor Arthur Schoenstadt 1
Code MA/Zh
Department of Mathematics
Naval Postgraduate School
Monterey, CA 93943

Professor H.B. Keller 1
Department of Applied Mathematics
California Institute of Technology
Pasadena, CA 91125

INTEL Scientific Computers 1
15201 N.W. Greenbrier Pkwy.
Beaverton, OR 97006

Professor R. T. Williams 1
Code MR/Wu
Naval Postgraduate School
Monterey, CA 93943

Professor David Gottlieb 1
Brown University
Division of Applied Mathematics
Box F
Providence, RI 02012

Mike Carron 1
Advanced Technology Staff
Code CST
Naval Oceanographic Office
Stennis Space Center, MS 39522-5001

Professor Melinda Peng 1
Code MR/Pg
Naval Postgraduate School
Monterey, CA 93943

Rex Thanakij 3
MASPAR Computer Corporation
749 N. Mary Ave.
Sunnyvale, CA 94086